L4: Entry 1 of 2 File: USPT Jan 7, 2003

DOCUMENT-IDENTIFIER: US 6505211 B1
TITLE: Method for providing for persistence of java classes where the persistence semantics may be orthogonal to the class definition

Detailed Description Text (43):
This approach involves serializing the contents of the object 110 and storing the serialzed contents as a BLOB in the database 120. The Java.TM. class definition is then mapped to a structured type that is a subtype of a special structured type (known as SysJavaSerialized) that is always defined for the system. The definitions for these types are as below: CREATE TYPE SYSJAVASERIALIZED AS (CONTENT BLOB(2G)) MODE DB2SQL; CREATE TYPE SCHEMANAME.STNAME UNDER SYSJAVASERIALIZED INSTANTIABLE WITHOUT COMPARISONS NOT FINAL MODE;

Detailed Description Text (45):
FIG. 4A is a flowchart that illustrates the logic performed during bind-in according to the preferred embodiment of the present invention. During bind-out from the RDBMS 116 to the JVM 108, the FROM_SQL built-in transform function of the RDBMS 116 constructs an ObjectStream (that represents an instance of SCHEMANAME.STNAME), and passes it to the JDBC driver 112 (Step 400). The JDBC driver 112 reads type information from the ObjectStream to materialize the appropriate Java.TM. class as an object 110 (Step 402), and uses Java.TM. deserialization (the readObject( ) method) on an ObjectInputStream representing the "content" BLOB field of the ObjectStream (Step 404) to materialize the Java.TM. object 110 encoded in the "content" field (Step 406).

Detailed Description Text (47):
The advantage of this approach is that only a few modifications need be made in the JDBC driver 112 to achieve this functionality. The disadvantage of this approach is that such structured type instances 124 can only be accessed through Java.TM. User-Defined Functions (UDFs) (where they need to perform Java.TM. deserialization) or through the JDBC driver 112. Also, the system-generated accessor functions for the structured type cannot access the fields of the Java.TM. object as they are encapsulated opaquely into the BLOB attribute of the structured type (this is the attribute called "content"). However, considering that these represent Java.TM. classes 110 bought from third party vendors, i.e., classes 110 that cannot be modified by the user, it is questionable if this is a problem.

Detailed Description Text (52):
FIG. 5A is a flowchart that illustrates the logic performed during bind-out according to the preferred embodiment of the present invention. During bind- out, the FROM_SQL built-in transform function of the RDBMS 116 constructs an ObjectStream BLOB that represents a structured type instance 124, and passes it to the JDBC driver 112 (Step 500). The JDBC driver 112 reads type information from the ObjectStream, determines that the corresponding Java.TM. class 110 does not implement the SQLData interface, and constructs a SQLObjectInputStream object 110 with the remainder of the ObjectStream, i.e., BLOB (Step 502). The JDBC driver 112 uses the Java.TM. deserialization interface with the SQLObjectInputStream object 110 (Step 504) and then initializes the Java.TM. object 110 with the results of this deserialization (Step 506). Remember that the readXXX( ) methods of SQLObjectInputStream can read data in the ObjectStream format.

Current US Original Classification (1):
707/103Y

Current US Cross Reference Classification (1):
707/10

Current US Cross Reference Classification (2):
707/103X

Current US Cross Reference Classification (3):
707/104.1

Current US Cross Reference Classification (4):
707/4

CLAIMS:

16. The method of claim 15, wherein the mapping step comprises: constructing an ObjectStream from the structured type instance that maps the class definition; reading type information from the ObjectStream to materialize the appropriate class definition as an object, and using deserialization on an ObjectInputStream representing the BLOB to initialize the attributes of the object.

42. The system of claim 41, wherein the logic for mapping comprises logic for: constructing an ObjectStream from the structured type instance that maps the class definition; reading type information from the ObjectStream to materialize the appropriate class definition as an object, and using deserialization on an ObjectInputStream representing the BLOB to initialize the attributes of the object.

68. The method of claim 67, wherein the mapping step comprises: constructing an ObjectStream from the structured type instance that maps the class definition; reading type information from the ObjectStream to materialize the appropriate class definition as an object, and using deserialization on an ObjectInputStream representing the BLOB to initialize the attributes of the object.

Previous Doc Next Doc Go to Doc#